# Practice Session 6: Ansible automation tool

**Make sure that you have already gone through Lab-05**

Ansible is a simple automation engine that automates cloud provisioning, configuration management, application deployment, intra-service orchestration, and many other needs. It uses the YAML language that you have used in Lab-02, in the form of Ansible Playbooks, and that allows you to describe your automation jobs in a way that approaches plain English. The aim of this lab is to get acquainted with the Ansible automation tool and get to know its most commonly used commands.

### Introduction to Ansible

Ansible works by connecting to your nodes and pushing out small programs, called "Ansible modules" to them. These programs are written to be resource models of the desired state of the system. Ansible then executes these modules (over SSH by default) and removes them when finished.

### References

Referred documents and websites contain supportive information for the practice.

### Manuals

1. How Ansible works
2. Demo of this lab (from last year). This may not have very exact steps.

## Exercise 1. Installation and basic configuration of Ansible v2.12.

The goal of this task is to get acquainted with the installation of Ansible using pip and configure the ansible to manage other virtual machines remotely.

*Note: you can use your local machine, but we recommend using the existing virtual machine (k8s-controller VM) instead.*

### Ansible on `k8s-controller` VM

Ansible can be installed on Centos using `pip`

- The ansible installation is already carried out in *Lab 04 - Kubernetes*, So no need to install ansible again here.
- Login to `k8s-controller` VM

- Check ansible version
    - `ansible --version`

```
[centos@shiva-k8-controller ~]$ ansible --version
ansible [core 2.12.5]
  config file = None
  configured module search path = ['/home/centos/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /home/centos/.local/lib/python3.9/site-packages/ansible
  ansible collection location = /home/centos/.ansible/collections:/usr/share/ansible/collections
  executable location = /home/centos/.local/bin/ansible
  python version = 3.9.13 (main, Jun  9 2022, 00:00:00) [GCC 11.3.1 20220421 (Red Hat 11.3.1-2)]
  jinja version = 2.11.3
  libyaml = True
```

- List *all* the ansible hosts `ansible --list-hosts all`

  Sample Output:

```
[centos@shiva-k8-controller ~]$ ansible --list-hosts all
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not match
'all'
  hosts (0):
```

- List *localhost* hosts `ansible --list-hosts localhost`

  Sample Output:

```
[centos@shiva-k8-controller ~]$ ansible --list-hosts localhost
[WARNING]: No inventory was parsed, only implicit localhost is available
  hosts (1):
    localhost
```

# Exercise 2. Basic Ansible commands

The aim of this task is to make you familiar with the basic ansible commands.

## 2.1. Ansible Hosts/inventories file

**Inventories**: Ansible works against multiple managed nodes or “hosts” in your infrastructure at the same time, using a list or group of lists known as inventory. You can specify a different inventory file at the command line using the `-i <path>` option. For more information on inventory follow here...

Try executing the `ansible --version`. Sample output of `ansible --version` command:

```
[centos@k8s-controller-chinmaya ~]$ ansible --version
ansible [core 2.12.5]
  config file = None
  configured module search path = ['/home/centos/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /home/centos/.local/lib/python3.9/site-packages/ansible
  ansible collection location = /home/centos/.ansible/collections:/usr/share/ansible/collections
  executable location = /home/centos/.local/bin/ansible
  python version = 3.9.9 (main, Nov 22 2021, 00:00:00) [GCC 11.2.1 20211019 (Red Hat 11.2.1-6)]
  jinja version = 2.11.3
  libyaml = True
```

From `ansible --list-hosts all` and `ansible --list-hosts localhost` commands, `all` and `localhost` are some groups of machines with related information that are expected to be present in default inventory file.

- Create a directory `mkdir ~/inventory && cd inventory` and Now let's create **hosts.yaml** file `vi hosts.yaml` and copy the following content. Make sure that you are in the `k8s-controller` VM.

```
[k8s-master]
master ansible_host=172.17.90.157 ansible_port=22 ansible_user=centos
ansible_ssh_private_key_file=/home/centos/.ssh/chinmayadehury.pem

[k8s-workers]
worker1 ansible_host=172.17.88.115 ansible_port=22 ansible_user=centos
ansible_ssh_private_key_file=/home/centos/.ssh/chinmayadehury.pem

worker2 ansible_host=172.17.88.115 ansible_port=22 ansible_user=centos
ansible_ssh_private_key_file=/home/centos/.ssh/chinmayadehury.pem
```

- Modify with your key name instead of **chinmayadehury**
- Change the IPs of **master** and **workers** with k8s-master and k8s-worker1, k8s-worker2 (highlighted above)
- **[k8s-master]** is the custom tag used to group your VM(s) and it can be changed.
- Here, **hosts.yaml** file contains the required information such as username or private_key or port are used to connect to the other VMs (i.e. **k8s-master and k8s-worker1, k8s-worker2**) over ssh.

*note: Writing `$HOME` instead of `/home/centos` in the `hosts.yaml` inventory file may not work.*

- The **hosts.yaml** can also have the hostname or the IP address of other VMs. Use can use either of one. An example of minimal version of an inventory file would look like below:

```
[iotusecase]
172.17.90.248
172.17.91.168
172.17.89.193
```

## 2.2. Basic tasks using Ansible

The `ansible` command defines and runs a single task **playbook** against a set of hosts. The `ansible-doc <module-name>` is a handy command to know more about a specific module.

- Now let's ping all servers in `k8s-master` and `k8s-worker` group
  `ansible -m ping k8s-master -i hosts.yaml`
  Sample Output:

```
[centos@shiva-k8-controller ~]$ ansible -m ping k8s-master -i hosts.yaml
[WARNING]: Invalid characters were found in group names but not replaced, use -vvvv to see details
master | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

  **NOTE**: `ping` module is useful from `/usr/bin/ansible` to verify the ability to login and that a usable Python is configured. This is NOT ICMP ping, this is just a trivial test module that requires Python on the remote-node.

  More about `ping` module:

https://docs.ansible.com/ansible/latest/collections/ansible/builtin/ping_module.html

- Execute `pwd` command in all servers in `k8s-workers` group
  `ansible -m command -a "pwd" k8s-workers -i hosts.yaml`
  Sample Output:

```
[centos@shiva-k8-controller ~]$ ansible -m command -a "pwd" k8s-workers -i hosts.yaml
[WARNING]: Invalid characters were found in group names but not replaced, use -vvvv to see details
worker2 | CHANGED | rc=0 >>
/home/centos
worker1 | CHANGED | rc=0 >>
/home/centos
```

  **NOTE**: The given command with `-a "command name"` option will be executed on all selected nodes.

  More about `command` module:
  https://docs.ansible.com/ansible/latest/collections/ansible/builtin/command_module.html

- Execute `ls` command in all servers in `k8s-workers` group
  `ansible -m command -a "ls" k8s-workers -i hosts.yaml`
  Sample Output:

```
[centos@shiva-k8-controller ~]$ ansible -m command -a "ls" k8s-workers -i hosts.yaml
[WARNING]: Invalid characters were found in group names but not replaced, use -vvvv to see details
worker2 | CHANGED | rc=0 >>
devops2022lab06
devops2022lab06updated
docker-compose.yaml
gitlab-runner_amd64.rpm
hosts.yaml
install-app.yaml
kubespray
swagger
worker1 | CHANGED | rc=0 >>
data
```

- Execute `ls` command in all servers in `k8s-workers` group with a private key. Change the private key name according to your ssh key name with **chinmayadehury**
  `ansible -m command --private-key .ssh/chinmayadehury -a "ls" k8s-workers -i hosts.yaml`
  Sample Output:

```
[centos@shiva-k8-controller ~]$ ansible -m command -a "ls" k8s-workers -i hosts.yaml
[WARNING]: Invalid characters were found in group names but not replaced, use -vvvv to see details
worker2 | CHANGED | rc=0 >>
devops2022lab06
devops2022lab06updated
docker-compose.yaml
gitlab-runner_amd64.rpm
hosts.yaml
install-app.yaml
kubespray
swagger
worker1 | CHANGED | rc=0 >>
data
```

- Create a directory in the remote VM using `command` module:
  `ansible -m command -a "mkdir /home/centos/mydir1" k8s-workers -i hosts.yaml`

**NOTE**: here `k8s-workers` is a hosts group.
Sample Output:

```
[centos@lab01-controller inventory]$ ansible -m command -a "mkdir /home/centos/mydir1" k8s-workers -i hosts.yaml
[WARNING]: Invalid characters were found in group names but not replaced, use -vvvv to see details
worker1 | CHANGED | rc=0 >>
```

- Create a directory in the remote VM using `file` module:
  `ansible -m file -a "path=/home/centos/mydir2 state=directory" k8s-workers -i hosts.yaml`
  **NOTE**: Here again I am using a `k8s-workers` host group (similar to `iotusecase`).
  Sample Output:

```
[centos@lab01-controller inventory]$ ansible -m file -a "path=/home/centos/mydir2 state=directory" k8s-workers  -i hosts
.yaml
[WARNING]: Invalid characters were found in group names but not replaced, use -vvvv to see details
worker1 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": true,
    "gid": 1000,
    "group": "centos",
    "mode": "0775",
    "owner": "centos",
    "path": "/home/centos/mydir2",
    "secontext": "unconfined_u:object_r:user_home_t:s0",
    "size": 6,
    "state": "directory",
    "uid": 1000
}
```

  More about `file` module:
  https://docs.ansible.com/ansible/latest/collections/ansible/builtin/file_module.html

- Transfer a file from local VM to the remote VM using `copy` module. Before this create a simple text file in the home directory `echo "Hi! Welcome to Ansible World" > ~/sample.txt` and than following command.
  `ansible -m copy -a "src=/home/centos/sample.txt dest=/home/centos/" k8s-workers -i hosts.yaml`
  Sample Output:

```
[centos@lab01-controller inventory]$ ansible -m copy -a "src=/home/centos/sample.txt dest=/home/centos/" k8s-workers -i
hosts.yaml
[WARNING]: Invalid characters were found in group names but not replaced, use -vvvv to see details
worker1 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": true,
    "checksum": "0b3a934763e8e4c6582305c77926028a26de095e",
    "dest": "/home/centos/sample.txt",
    "gid": 1000,
    "group": "centos",
    "md5sum": "aee78cfad1e2458238160672cf18d152",
    "mode": "0664",
    "owner": "centos",
    "secontext": "unconfined_u:object_r:user_home_t:s0",
    "size": 29,
    "src": "/home/centos/.ansible/tmp/ansible-tmp-1665451598.5953383-47813-17539238600786/source",
    "state": "file",
    "uid": 1000
}
```

**Screenshot - 1**

- Take a screenshot of the output of the command *history* that includes all your executed ansible commands from the above tasks.

## Exercise 3. Working with Ansible Playbook

Ansible command line is great for executing a single task. But playbooks are more useful for multiple tasks. Playbooks are text files written in the YAML format. You can check http://www.yamllint.com/ or other alternatives to validate your YAML file.

The basic command to invoke any playbook is:
`ansible-playbook <path to your playbook>`

Make sure that you are in the k8s-controller VM.
More about `ansible-playbook` command:
https://docs.ansible.com/ansible/latest/cli/ansible-playbook.html

### 3.1. Playbook for file/directory operations

Following playbook will let you execute the above Ansible commands, e.g create new directory, new files, copy file from local machine to remote machine, print the message, etc.

Create your first playbook with following content.

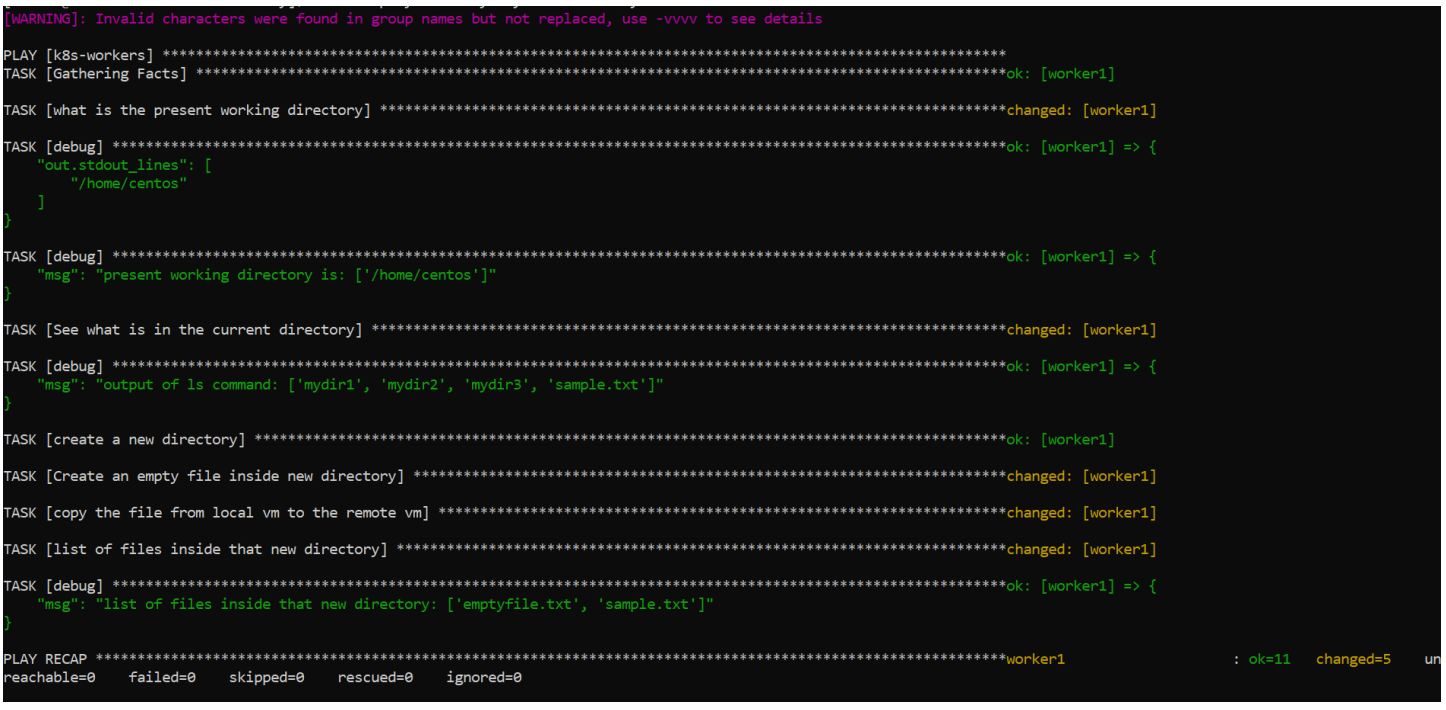
`vi play-1.yaml`

```
---
- hosts: remote

  tasks:
   - name: what is the present working directory
     command: pwd
     register: out
   - debug:
      var: out.stdout_lines
   - debug:
      msg: "present working directory is: {{out.stdout_lines}}"

   - name: See what is in the current directory
     command: ls
     register: lsout
   - debug:
      msg: "output of ls command: {{lsout.stdout_lines}}"

   - name: create a new directory
```

```
    file:
      path: /home/centos/mydir3
      state: directory

  - name: Create an empty file inside new directory
    file:
      path: /home/centos/mydir3/emptyfile.txt
      state: touch

  - name: copy the file from local vm to the remote vm
    copy:
     src: /home/centos/sample.txt
     dest: "/home/centos/mydir3/"

  - name: "list of files inside that new directory"
    shell: "ls /home/centos/mydir3/"
    register: lsout
  - debug:
     msg: "list of files inside that new directory: {{lsout.stdout_lines}}"
```

NOTE: Update the *remote* group name in the second line of above content ( `hosts: remote` ) based on your inventory file and host group name. Here, you can use either `hosts:k8s-master` or `hosts:k8s-workers`

Ansible playbook command: `ansible-playbook play-1.yaml -i hosts.yaml`
Sample Output:




You may visit below source for more information

- https://docs.ansible.com/ansible/2.8/modules/list_of_files_modules.html
- https://docs.ansible.com/ansible/2.8/modules/file_module.html#file-module

## 3.2. Package install/remove/reinstall/download operation

Below is a playbook example for working with package management (using `yum` module).

*Note: Following things will work only on the centos Operating system. DIY: find out why.*

`vi install-packages.yaml`

```
---
- hosts: remote
  gather_facts: true
  become: true
  become_method: sudo

  tasks:
    - name: Install the latest version of vim editor
      yum:
        name: vim
        state: latest

    - name: Download the nano editor package but do not install it
      yum:
        name:
          - nano
        state: latest
        download_only: true
        download_dir: "/home/centos/"
    - name: Install the latest version of Apache
      yum:
        name: httpd
        state: latest
    - name: Start apache service
      service:
        name: httpd
        state: started
```

NOTE: Update the *remote* group name in the second line of above content ( `hosts: remote` ) based on your inventory file and host group name. Here, you can use either `hosts:k8s-master` or `hosts:k8s-workers`

Command: `ansible-playbook install-packages.yaml -i hosts.yaml`

Sample Output:

```
[centos@lab4-ansible-test-chinmaya playbooks]$ ansible-playbook install-packages.yaml

PLAY [remote] ******************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************
ok: [lab4-ansible-test-chinmaya-2]

TASK [Install the latest version of vim editor] ********************************************************************
ok: [lab4-ansible-test-chinmaya-2]

TASK [Download the nano editor package but do not install it] ******************************************************
changed: [lab4-ansible-test-chinmaya-2]

TASK [Install the latest version of Apache] ************************************************************************
ok: [lab4-ansible-test-chinmaya-2]

TASK [Start apache service] ****************************************************************************************
ok: [lab4-ansible-test-chinmaya-2]

PLAY RECAP *********************************************************************************************************
lab4-ansible-test-chinmaya-2 : ok=5    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Now go to the browser (You should use IP address to access based on the host group name you mentioned ) and test if apache web server is accessible, like below:

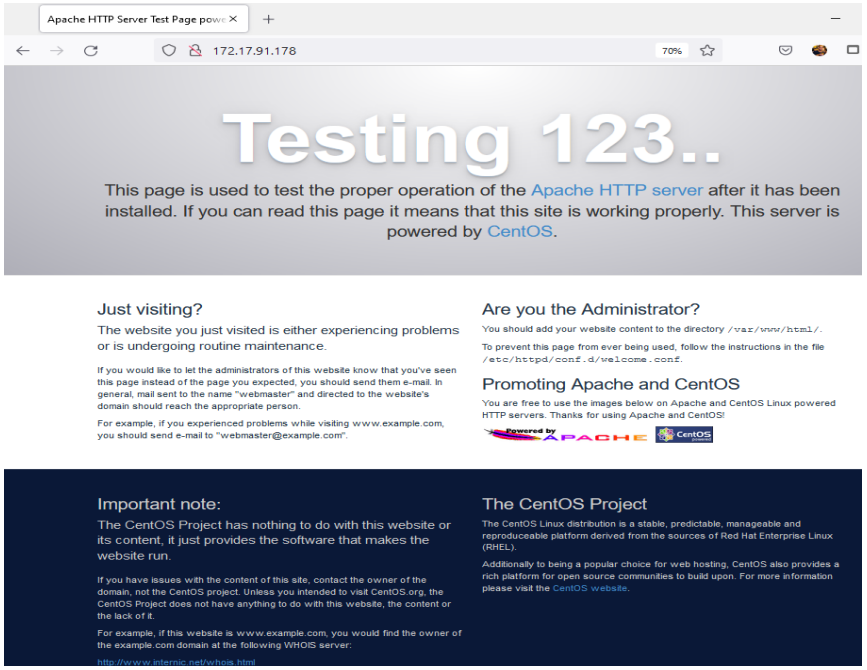



Figure out if you are unable to access the apache test page.

**Hint:**

- login to other VM(s) where you have installed the latest version of Apache and check if the httpd server is running
- Check the security group of the remote VM, if required.

**Screenshot - 2**

- Take a screenshot of output of the Apache server running in your browser and IP should be visible.

## 3.3. Playbook to stop and remove a package

Below playbook will stop the previously started apache server and remove/uninstall from the system *Note: Following playbook will work only on the centos Operating system (DIY: find out why).*

`vi remove_package.yaml`
==========

```
---
- hosts: k8s-workers
  gather_facts: true
  become: true
  become_method: sudo

  tasks:
    - name: Stop service httpd, if started
      service:
        name: httpd
        state: stopped
    - name: Remove apache package
      yum:
        name: httpd
        state: absent
```

NOTE: Update the *remote* group name in the second line of above content ( `hosts: remote` ) based on your inventory file and host group name. Here, you can use either `host:k8s-master` or `host:k8s-workers`

Command: `ansible-playbook remove_package.yaml -i hosts.yaml`
Sample Output:

```
[centos@lab4-ansible-test-chinmaya playbooks]$ ansible-playbook remove_package.yaml

PLAY [remote] ******************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************
ok: [lab4-ansible-test-chinmaya-2]

TASK [Stop service httpd, if started] ******************************************************************************
changed: [lab4-ansible-test-chinmaya-2]

TASK [Remove the Apache package] ***********************************************************************************
changed: [lab4-ansible-test-chinmaya-2]

PLAY RECAP *********************************************************************************************************
lab4-ansible-test-chinmaya-2 : ok=3    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

# Exercise 4. Deploy flask application on k8s-cluster

In this task, you will use Dockerized flask application image from **Practice Session 2** , **Exercise 4**. and deploy on the k8s-cluster using ansible playbooks. Before this, we will try to deploy ngnix web server on k8s-cluster using ansible playbook.

## 4.1. Deploy your nginx service on k8 cluster

Create a playbook with following content.

vi install-app.yaml
==========

```
---
- name: create
  hosts: k8s-master
  gather_facts: true
  become: true
  become_user: centos

  tasks:
    - name: Create target directory
      file:
       path=/home/centos/deploy_nginx/
       state=directory

    - name: create the deployment file
      lineinfile:
            path: /home/centos/deploy_nginx/deploy1.yaml
            create: yes
            line: |
              apiVersion: apps/v1
              kind: Deployment
              metadata:
                name: nginx-deployment
                labels:
                  app: nginx
              spec:
                replicas: 1
                selector:
                  matchLabels:
                    app: nginx
                template:
                  metadata:
                    labels:
                      app: nginx
                  spec:
                    containers:
                    - name: nginx
                      image: nginx:1.14.0
                      ports:
                      - containerPort: 80
    - name: create the deployment by running the kubectl command
      command: "kubectl create -f /home/centos/deploy_nginx/deploy1.yaml"

    - name: create the service file
```

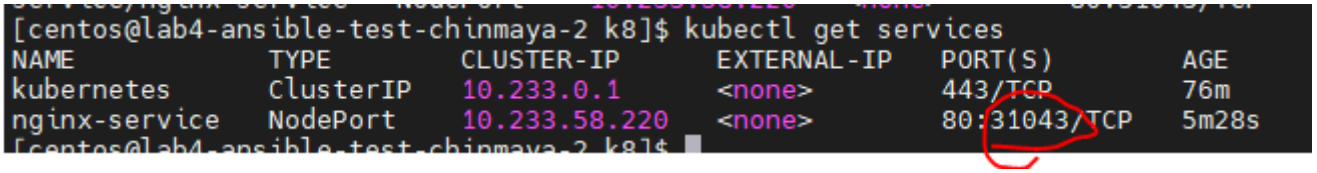
```
    lineinfile:
          path: /home/centos/deploy_nginx/service1.yaml
          create: yes
          line: |
            apiVersion: v1
            kind: Service
            metadata:
                  name: nginx-service
                  labels:
                    run: nginx-service
            spec:
                  type: NodePort
                  ports:
                  - port: 80
                    protocol: TCP
                  selector:
                    app: nginx
  - name: create the service
    command: "kubectl create -f /home/centos/deploy_nginx/service1.yaml"
```

*Note: Make sure that the alignment (space and tab character) is correct.*

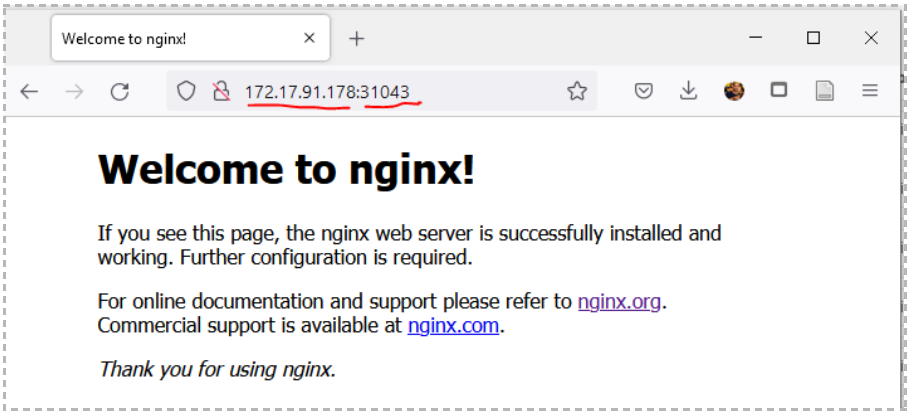
- Now execute the playbook:
  `ansible-playbook install-app.yaml -i hosts.yaml`

After Nginx app deployment, go to the master node and execute `kubectl get services` command to know the port number. (Here we assume that you have finished “Lab04: Working with Kubernetes” and you are using the same VMs to perform this exercise.)

Get the port number (as shown in below figure)




Now in the browser hit master node's external_ip with the obtained port number (as shown in above fig). Output may look like below.




**Screenshot - 3**

- Take a screenshot of output of the nginx server running in your browser and IP should be visible.

## 4.2. Deploy flask application on k8 cluster

In this task, you're going to deploy the flask application from **Practice Session 2** and **Exercise 4**. This task needs to be completed by yourself with minimum instructions noted below.

- Create the ansible playbook with name **install-flask.yaml and i**t should be similar to **install-app.yaml**
- Add `name`, `hosts`, `gather_facts`,`become`,`become_user`
- Add the first task, i.e. `Create target directory` task to create directory `deploy_flask`,
- Create the second task to create `pv.yaml` *file,* as shown below

```
- name: create the persistent volume file
  lineinfile:
        path: /home/centos/deploy_flask/pv.yaml
        create: yes
        line: |
          apiVersion: v1
          kind: PersistentVolume
          metadata:
            name: pvolume
            labels:
              type: local
          spec:
            storageClassName: manual
            capacity:
              storage: 5Gi
            accessModes:
            - ReadWriteOnce
            hostPath:
              path: "/mnt/data"
```

```
              nodeAffinity:
                required:
                  nodeSelectorTerms:
                  - matchExpressions:
                    - key: kubernetes.io/hostname
                      operator: In
                      values:
                      - node1
```

- Add the third task to create persistent volume using kubectl command

```
 - name: create the persistent volume
   command: "kubectl create -f /home/centos/deploy_flask/pv.yaml"
```

- Add fourth task, i.e. *create the deployment file* task as same as in **install-app.yaml** from the above exercise (Exercise 4.1), with following hints.

```
- name: create the deployment file
   lineinfile:
         path: /home/centos/deploy_flask/deploy1.yaml
         create: yes
         line: |
           apiVersion: apps/v1
           kind: Deployment
           metadata:
             name: flask-deployment
             labels:
               app: flask
           spec:
             replicas: 1
             selector:
               matchLabels:
                 app: flask
             template:
               metadata:
                 labels:
                   app: flask
               spec:
                 containers:
                 - name: flask
                   image: shivupoojar/testprac2
                   ports:
```

```
            - containerPort: 5000
            lifecycle:
              postStart:
                exec:
                  command: ["cp", "/data/CO2.csv", "/app/CO2.csv"]
            volumeMounts:
            - mountPath: "/data"
              name: data-storage-volume
          volumes:
            - name: data-storage-volume
              persistentVolumeClaim:
                claimName: vclaim
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: vclaim
  namespace: default
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: manual
```

- Update `image` with your flask app image name
- Update fifth task, i.e. *create the deployment by running the kubectl command* task.
    - Update the path to the yaml. The path should be the same as the path of the previous (or second) task.
- Update sixth task, i.e. *create the service file task.*
    - Update `path` value,
    - Update service `name`, as per your choice (name:service-flask,)
    - Update port and targetPort number from 80 to 5000
    - Update `selector` i.e `app:flask` as well
- Update the kubectl command accordingly, in the seventh task, i.e. *create the service* task.
- Run the playbook `ansible-playbook install-flask.yaml -i hosts.yaml`

Finally, when the playbook is executed (on k8s-master) you should see the sensor data on the web page.

**Screenshot - 4**

Take a screenshot of a webpage where your name and IP address are clearly seen.

# Exercise 5. Working with gitlab, ansible and k8s-cluster

In this task, you're going to automate the deployment of an application on k8s-cluster using gitlab and ansible.

- Create a project in the gitlab with name `lab06-Ansible_k8s_deployment` under group `Devops2022fall/students/devops2022Fall-<lastname>-<studyCode>`
- Install the gitlab-runner on k8s-controller virtual machine
  - Download the gitlab runner - `curl -LJO "https://gitlab-runner-downloads.s3.amazonaws.com/latest/rpm/gitlab-runner_amd64.rpm"`
  - Install the runner: `sudo rpm -i gitlab-runner_amd64.rpm`
  - Get the token for Gitlab runner registration
    - Go to your `lab06-Ansible_k8s_deployment` Gitlab project
    - Go to `Setting --> CI/CD → Expand Runner`
    - Note down Registration token
  - Registering the Gitlab runner to your project.
    - **Configure** the runner using `sudo gitlab-runner register` command.

      Provide the following details
      - Enter the GitLab instance URL: https://gitlab.cs.ut.ee/
      - Enter the registration token: as noted in the earlier step
      - Enter a description for the runner.(ex. ansible)
      - Enter tags for the runner (comma-separated): `build,deploy`
      - Enter optional maintenance note for the runner: Press Enter
      - Enter an executor: It should be `docker` (not `shell`)
      - You can provide a default docker image, e.g. `ubuntu:18.04`
    - Now you can see the registered runner in your gitlab account in `Settings → CI/CD → Runners.`
    - Make sure that you have disabled "`Enable shared runners for this project`" option, under `Settings → CI/CD → Runners` option, else your job might get assigned to the shared runner.
    - Sample runner:


- Upload the ansible-playbook file **install-flask.yaml (**you created in previous Ex 4) **,** Inventory file **hosts.yaml and ssh key file** to `lab06-Ansible_k8s_deployment` Gitlab project.
- Create a `.gitlab-ci.yml` file (the way you did in Lab04 ) and add the ansible playbook command you used in the previous exercise to the "*scripts"* key..
  It should look like:
  Change the seatis.pem to your ssh key file name

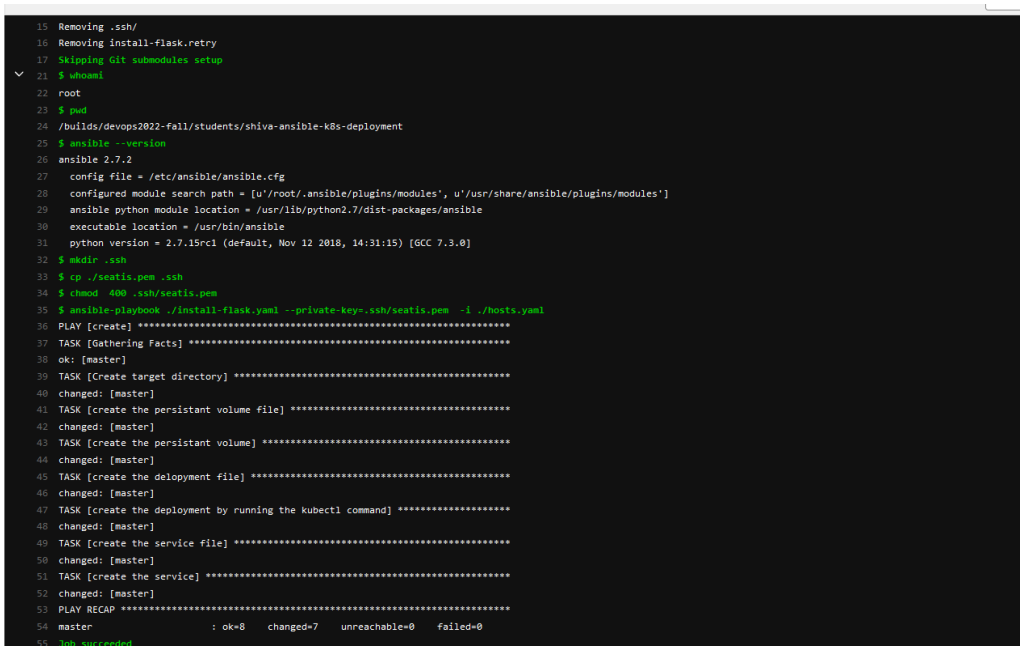
```
stages:
  - deploy

deploy:
  stage: deploy
  image:
    name: williamyeh/ansible:ubuntu18.04
  variables:
    ANSIBLE_HOST_KEY_CHECKING: "False"

  script:
    - mkdir .ssh
    - cp ./seatis.pem .ssh
    - chmod  400 .ssh/seatis.pem
    - ansible-playbook ./install-flask.yaml --private-key=.ssh/seatis.pem  -i
./hosts.yaml

  allow_failure: true
```

If you get error related to private key file while executing pipeline, than modify the path of ssh key file like ansible_ssh_private_key_file=.ssh/seatis.pem (seatis.pem should be replaced with your key file name) in the hosts.yaml in your gitlab project

Finally, when the playbook is executed (on k8s-master) you should see the sensor data on the web page

## Screenshot - 5

Take a screenshot of an executed pipeline like shown below.



# Deliverables

1- Gather all the screenshots

- Screenshot 1
- Screenshot 2
- Screenshot 3
- Screenshot 4
- Screenshot 5

2- zip the all the ansible-playbook and inventory(hosts.yaml) files from Exercise 3, 4 and 5 and all the screenshots.

3- Upload the zip file to the course wiki page.

4- You may **Stop** the Virtual Machines and you can start using the same in the next **practice session.**

**Don't delete your VMs**